أساسيات وتطبيقات لغة سي

الجزء الثالث

معلومات عن المؤلف

الدكتور **عمر زرتي** هو أستاذ شرف بقسم الحاسب الآلي بكلية العلوم جامعة الفاتح بطرابلس ليبيا.

تحصل على شهادة البكالوريوس سنة 1970 في الرياضيات من جامعة Colorado School of Mines الواقعة في مدينة Golden من ولاية كولورادو بالولايات

المتحدة. ثم نال شهادة الماجستير سنة 1972 من جامعة Case Western Reserve الكائنة بمدينة Cleveland بولاية أوهايو . ثم تحصل على شهادة الدكتوراة سنة 1976 من جامعة Washington State University بمدينة Pullman بولاية واشنطن.

عند رجوعه الى جامعة الفاتح قام بتأسيس قسم الحاسب الآلي وترأس القسم لمدة خمس سنوات، واستمر بعده كعضو هيئة تدريس في مجال (الطرق العددية والبرمجة) ابتداء من درجة (محاضر) وانتهاء إلى درجة (أستاذ).

وقد اتجه إلى مجال تأليف وكتابة الكتب المنهجية في المستوى الجامعي والثانوي. ومن أهم كتبه كتاب (الطرق العددية بلغة فورتران) ، كما يعتبر كتابه (أساسيات الحاسوب والبرمجة) من الكتب التي لاقت استحسانا وقبولا واسعا. إضافة إلى ذلك فقد قام بتأليف كتاب (البرمجة بلغة باسكال) و (البرمجة بلغة فورتران) و(أساسيات وتطبيقات لغة سي) ، وتمت طباعة هذه الكتب عدة مرات.

وللدكتور عمر زرتي أيضا اهتمام بموضوع (بحوث العمليات) الذي يقوم بتدريسه في مستوى الدراسات العليا، ويقوم بالاشراف على رسالة ماجستير في هذا المجال.

الباب السادس

6

المصفوفات والنضائد

Arrays & Strings

6.1 مقدمة

المصفوفة هي مجموعة من العناصر المتجانسة تحت اسم واحد، حيث يمكن تمييز كل عنصر بترتيبه (أي دليله) في المصفوفة . فمثلاً مجموعة الأعداد :

$a_0 = 5.1$
$a_1 = 6.3$
$a_2 = 7.4$
....
$a_9 = 8.2$

تكوّن مصفوفة من 10 عناصر من النوع الكسري float .
افترض أننا نريد قراءة هذه العناصر وإيجاد متوسطها ثم إيجاد الفرق بين كل عنصر والمتوسط.
ستوضح لنا هذه المسألة أهمية المصفوفات . إذا اعتبرنا الطريقة العادية، حيث نرمز للمدخلات بمتغير واحد وليكن x ، فإن حساب المتوسط يتم بسهولة على النحو التالي :

جامعة الفاتح –كلية العلوم – طرابلس – ليبيا
omarzarty@yahoo.com

```
 for( k = 1 , sum = 0  ; k < =10 )
{  scanf ( "%f "  , &x )  ;
   sum = sum +  x  ;
}
average  =  sum / 10  ;
```

ولكن الآن لحساب الفرق بين كل عنصر والمتوسط ، نجـد أن علينـا قـراءة البيانات مرة أخرى، حيث لا يوجد لدينا في ذاكرة الحاسوب إلا آخر قيمة تمت قراءتها للمتغير x .

لذلك فإن الفكرة الأساسية وراء مفهوم المصفوفة array هي تخزين كل عناصر المصفوفة في الذاكرة الرئيسية والرجوع إليها وقت الحاجة .

هذا يتطلب طبعاً التمييز بين عنصر وآخر، نظراً لأن جميع العناصـر تتـدرج تحت اسم واحد هو اسم المصفوفة ، فإن كل عنصر يتميز برقم صحيح يرمـز لترتيبه في المصفوفة، ويسمى هذا الرقم بالدليل index. فمثلاً العنـصر v [i] هو العنصر رقم i في المصفوفة v .

6.2 الحجز في الذاكرة

إذا أردنا استخدام مصفوفة في البرنامج ، فلابد في البداية من حجز عـــدد مـــن المواقع في الذاكرة يكفى لجميع عناصر هذه المصفوفة، ومـــع الحجــز يجـــب الإعلان عن نوع هذه العناصر حتى تخصص لها المواقع المناسبة لحجمها .

فإذا كانت المصفوفة cost ذات 7 عناصر من النوع الكسري ، فإن الإعـــلان عن ذلك يتم على النحو التالي:

```
float  cost [7] ;
```

لاحظ أن ذلك يعنى أن عناصر المصفوفة هي :

```
cost [0]
cost [1]
cost [2]
cost [3]
cost [4]
cost [5]
cost [6]
```

أي أن آخر عنصر يكون دليله أقل بواحد من الرقم المعلن عنه في الحـــــجز ، والسبـــب هو البداية من الصفر ، أي أن العنصر الأول دليله = 0 ، والعنـــصر الثاني دليله = 1 ، .... الخ .

مثـال : اكتب برنامجا لحساب متوسط دخل فرد خلال أسبوع (أي سبعة أيام) من دخله اليومي، ثم طباعة الفرق بين دخل كل يوم والمتوسط .

سبق وأن تطرقنا إلى هذا المثال في المقدمة للمصفوفات ، ورأينا أن عدم استخدام المصفوفات يؤدى إلى ضرورة قراءة البيانات مرتين .
طبعاً يمكننا في هذا المثال تخزين دخل الفرد في كل يوم على النحو :

income0 , income1 , income2 , ..., income 6

أي بدون استخدام المصفوفات، ولكن ماذا لو كان المطلوب المتوسط الشهري أو حتى السنوي ؟ بكل تأكيد ستكون قائمة الأسماء طويلة جدا.
لذلك نستخدم الإعلان

```
float income [7] ;
```

لنعلن عن مصفوفة ذات 7 عناصر كسرية اسمها income .

بعد قراءة هذه العناصر وحساب المتوسط نستطيع الرجوع إليها لحساب الفرق diff بين كل عنصر والمتوسط كما هو مبين بالبرنامج (6.2.1) .

```
main()
{       float income[7], sum , average;
        int i;
        for(i=0; i<=6 ; i++)
        {
          printf("\n Enter income for day[%d]"
                            ,i+1 );
          scanf("%f",&income[i]);
          sum += income[i];
        }
        average=sum/7;
        printf("\n avrage=%6.3f",average);
        printf("\n day        income         diff");
        for(i=0; i<=6 ; i++)
           printf("\n %d      %6.2f  %6.2f",
        i+1,income[i],income[i]-average);
}
```

الشكل (6.2.1) برنامج حساب المتوسط والفرق بين كل عنصر والمتوسط

6.3 ترتيب المصفوفة Array Sorting

من أهم تطبيقات المصفوفة أنها تمكننا من إعادة ترتيب عناصرها إمــا ترتيبــاً تنازلياً أو ترتيباً تصاعدياً .

هناك طبعاً عدة طرق لترتيب المصفوفات. مثلا لو أعطيت الأعداد التالية :

x [0] = 10
x [1] = 42
x [2] = 33
x [3] = 15
x [4] = 26

كيف ترتبها تنازلياً (أي ابتداء من أكبر قيمة إلى أصغر قيمة) ؟ هناك طبعا عدة طرق لاجراء هذا الترتيب. إحدى هذه الطرق مثلاً الخوارزمية التالية :

- من i=0 إلى i = n-2 (حيث n-1 هو دليل آخرعنصر في المصفوفة)

  إذا كانت x [i] < x [i+1] فاستبدل قيمتيهما .

- هل أصبحت المصفوفة مرتبة ؟

- إذا كانت الإجابة نعم توقف وإلا فارجع إلى الخطوة ❶ .

والبرنامج التالي يترجم هذه الخوارزمية إلى لغة سي:

```
/*----------------PROGRAM EX631.C -----------*/
#define N 5
main()
{       int x[N], i ,k, temp , sorted;
        for(i=0; i<N ; i++)
        {
```

```
            printf("\nEnter element[%d] ",i);
            scanf("%d",&x[i]);
      }
      for(k=0;;k++)
      {   sorted=1;
            for(i=0; i<N-1 ; i++)
            {
                  if(x[i+1] > x[i] )
                  {
                        sorted = 0;
                        temp = x[i];
                        x[i] = x[i+1];
                        x[i+1] = temp;
                  }
            }
            if(sorted) break;
      }
      printf("\n Sorted array after %d iter", k);
      for(i=0; i<N ; i++)
            printf("\n %d",x[i]);
}
```

الشكل (6.3.1) ترتيب البيانات

من الواضح في هذا البرنامج أن هناك حلقتين . الحلقة الخارجية تستمر بـــدون تحديد حتى يتم ترتيب المصفوفة ، والحلقة الداخلية تتم بعد n دورة.

من الملاحظ أيضا في البرنامج بالشكل (6.3.1) استخدام المتغير sorted الذي يبدأ بالقيمة 1 ثم يتحول إلى 0 إذا ما وجدنا عنصرين غير مرتبين ، وفى نهاية

الدورة الداخلية نختبر قيمة هذا المتغير ، فإذا كانت 1 فذلك يعنـــى أن عمليـــة الترتيب قد تمت كما يجب . ونخرج من الدورة الخارجية اللانهائية بالجملة :

```
if ( sorted ) break ;
```

وهى تكافئ الجملة :

```
if ( sorted = = 1) break ;
```

ولكن أكثر اختصاراً .

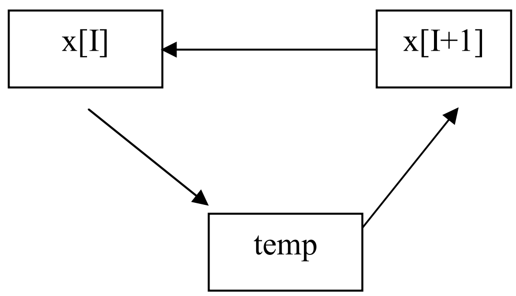
وثمة ملاحظة أخرى جانبية عن هذا البرنامج وهي أنه يطبع عـــدد الـــدورات iterations التي أداها للوصول إلى حل (أي إلى المصفوفة المرتبة تنازليـــا) . طبعاً هذا العدد يختلف حسب المصفوفة المدخلة ، ولكنه في أســـوأ الأحـــوال لا يزيد عن N (عدد العناصر) .

كان بالإمكان أن نكتب الدورة الخارجية على النحو :

```
for( k=0 ; k<N ; k++) { ……}
```

والاستغناء عن اختبار المتغير sorted ، والحلقة اللانهائية، ولكن قد لا نحتـــاج أحياناً إلى N من الدورات بل أقل من ذلك ، وبالـــذات إذا كانـــت المـــصفوفة المدخلة تكاد تكون مرتبة تنازليا ، لذلك استخدمنا المتغير sorted للخروج مـــن الدورة الخارجية بمجرد الوصول إلى المصفوفة المرتبة.

لاحظ أيضاً أن البرنامج يقوم بعملية اســتبدال x [i] مــع x [i+1] بالطريقــة المعتادة في البرمجة ، وهى تتم عن طريق وسيط مؤقت temp نضع فيه x [i] قبل وضع x[i+1] في x [i] على الشكل التالي :



الشكل (6.3.1) استبدال x [i] مع x [i+1]

6.4 النضائد Strings

النضيد هو عبارة عن مصفوفة من الرموز . وهذه الرموز تــشمل الحــروف الهجائية ، والأرقام والإشارات ، المبينة بجدول آسكى ascii (أنظر الملحق) .
في لغة سى ، يوضع النضيد بين علامات التنصيص المزدوجة مثل :

" Kamal Ahmed "
" 32 Cairo Street "

لاحظ أن هذه الرموز (سواء أكانت حروف أم أرقامــاً) تخــزن فــي ذاكــرة الحاسوب كأرقام ثنائية وذلك حسب الشفرة المستخدمة، لذلك يمكن أن ننظر إلى

النضيد على أنه مصفوفة من الأعداد الصحيحة. وهذه الأعـــداد لا يتجاوز كل منها العدد الذي يحمله الحـيز المـخـصص لحرف واحد في الحاسوب ، وهو عادة بايت واحدة أي 8 بت. عند تخصيص حيز في الذاكرة للنضيد يجب الانتباه إلى إضافة موقع واحد لرمز نهاية النضيد (ويرمز له بالرمز '\0') ، فإذا كـان النضيد المطلوب هو :

"UNIVERSITY"

فيجب الإعلان عن 11 رمز لهذا النضيد (10 أحرف بالإضـافة إلـى رمـز الانتهاء) ، ويكون الإعلان على النحو التالي :

```
char  w [11]  ;
```

حيث يرمز المتغير w لمصفوفة عناصرها من النوع الرمـزي char ذات 11 عنصر منها 10 رموز عادية ورمز الانتهاء '\0'

<u>مثال</u> : اكتب برنامجا يقوم بقراءة اسم المستخدم ، ولقبه (بحيـث لا يزيـد الاسم عن 11 حرفا ولا يزيد اللقب عن 14 حرفا) ويطبع الآتي:

Your name is ....

حيث يطبع الاسم واللقب في هذا السطر مع وضع فراغ واحد بينهما.

لدينا هنا مصفوفتان الأولى للاسم ولتكن name، والأخرى للقب ولتكن family . الأولى تحتاج إلى 12 رمزاً (11 حرفاً مع رمز الانتهاء) والثانية تحتاج إلـى 15 رمزاً (14 حرفاً مع رمز الانتهاء) .

```
main()
{
        char name[12];
        float family[15];
        printf("\n What is your first name? ");
        scanf("%s",name);
        printf("\n What is your family name?" );
        scanf("%s",family);
        printf("\n Your name is %s %s ", name, family);
}
```

الشكل (6.4.1) قراءة نضيدين وطباعتهما في سطر واحد

ملاحظات

❶ لاحظ الآن أن دالة القراءة scanf تستخدم الرمز s% لقراءة النضيد (حيث s ترمز لكلمة string نضيد) .

❷ لاحظ أيضاً عند قراءة نضيد ما إمكانية الاستغناء عن الرمز & الذي يوضع عادة قبل المتغير المطلوب قراءته في دالة scanf ، وسيتضح السبب فيما بعد .

❸ توجد في لغة سى دالة أخرى غير دالة scanf لقراءة النضيد وهى gets (اختصار get string) ، أي كان بالإمكان كتابة الجملة :

gets (name) ;

بدلًا من الجملة:

scanf (" %s " , name) ;

جامعة الفاتح –كلية العلوم – طرابلس – ليبيا
omarzarty@yahoo.com

كما في المثال التالي:

<u>مثال</u> : اكتب برنامجا لقراءة نضيدين string2 و string1 لا يزيد طول كل منهما عن 20 رمزاً، وتكوين نضيد ثالث string3 يتكون من دمج النضيدين بحيث يأتي string2 في نهاية string1 .

يبين الشكل (6.4.2) البرنامج المطلوب وفيه نلاحظ ما يلي:

1 – استخدم الدالة gets لقراءة نضيد

2- استخدام الاختبار

string [i] = = '\0'

لاختبار نهاية النضيد string

3- بعد وضع النضيدين في النضيد string3 يجب إنهاء النضيد الجديد بوضع '\0' في نهايته ، بواسطة الجملة :

```
string3 [k]  =  '\0' ;
```

4- إمكانية طباعة نضيد بواسطة الدالة puts مثل :

puts (string3) ;

5- عملية ربط نضيدين في نضيد واحد من المسائل المعروفة، ويطلق عليها عادة مصطلح concatenation، ويوجد لها في لغة سى دالة جاهزة اسمها strcat .

لذلك كان من الممكن اختصار البرنامج (6.4.2) باستخدام هذه الدالة على النحو التالي:

strcat (string1 , string2) ;

وهى جملة تقوم بإضافة string2 إلى string1 .

```
main()
{
        char string1[20] , string2[20] , string3[40] ;
        int i , k,j ;
        printf("\n enter string1-->");
        gets( string1 );
        printf("\n enter string2-->");
        gets( string2 );
        for(i=0; ; i++)
        {       if( string1[i] == '\0' ) break;
                string3[i]=string1[i];
        }
        for(k=i, j=0 ;   ; k++,j++)
        {       if( string2[j] == '\0' ) break;
                string3[k] = string2[j];
        }
```

```
        string3[k] = '\0' ;
        puts ( string3 );
}
```

الشكل (6.4.2) برنامج دمج نضيدين

وإذا أردنا نسخ string1 في string3 نستخدم الدالة strcpy (الكلمة مصدرها string copy) على النحو التالي :

```
strcpy ( string 3 , string 1)   ;
```

مثال : اكتب برنامجا لقراءة عدد صحيح يتكون من 5 خانات وطباعته مع التأكد من عدم وجود خطأ في الإدخال . أي أن البرنامج لا يقبل العدد المدخل حتى يتأكد من عدم وجود رمز غير الأرقام من 0 إلى 9 .

سوف نتبع في هذا البرنامج الخوارزمية التالية :

1. اقرأ الرقم الأول كرمز (دون إظهاره على الشاشة).
2. هل تم إدخال الرقم بصورة صحيحة ؟ أي هل يقع الرمز في الفئة ؟ { 0 , 1 , 2 , ... , 9 }
3. إذا كانت الإجابة (نعم) اكتب الرقم المدخل على الشاشة ، وانتقل إلى الرقم الذي يليه (إلى غاية 5 أرقام) .
وإلا ستستمر الحلقة لانهائية مع إصدار تنبيه صوتي بذلك.

هذه الخوارزمية نترجمها إلى برنامج بلغة سى في الشكل (6.4.3)، حيث نلاحظ ما يلي:

- الدالة getch (لاختصار get character) تقوم باستقبال رمز واحد من لوحة المفاتيح دون إظهاره على الشاشة.
- وبمناسبة ذكر هذه الدالة ، نلاحظ أن هناك دالة أخـــرى تـــؤدى نفـــس الغرض ولكن مع إظهار الرمز المدخل على الشاشة وهذه الدالة هـــي getche حيث تزيد بحرف e على الدالة getch.
- الدالة putchar (اختصار put character) تقوم بإظهار رمز واحـــد على الشاشة .

```
#include <stdio.h>
main()
{
        char x[5];
        int i;
        printf("\n Please enter a number-->");
        for(i=0;i<=4 ;i++)
                for(;;)
                {       x[i]=getch();
                        if( x[i]>= '0' && x[i]<= '9')
                        {    putchar(x[i]);
                                break;
                        }
```

```
                    else
                    printf("\a");
                }
        x[5]='\0';
        printf("\n The number entered is %s ", x);
        getch();
}
```

الشكل (6.4.3) التحقق من إدخال عدد صحيح

- بعد أن عينّا الخانات الخمس في العدد المطلوب وضعنا علامة نهاية النضيد في الخانة السادسة ، أي x [5] ، على النحو :

x [5] = '\0' ;

وذلك لغرض اكتمال النضيد x وطباعته باستخدام الدالة printf . كان من الممكن أيضاً طباعته بواسطة الدالة puts .

- نلاحظ أن الدالة getch التي وضعت في آخر البرنامج تهدف إلى الاحتفاظ بشاشة الإخراج حتى يتم إدخال أي رمز في لوحة المفاتيح keyboard .
- أخيراً نلاحظ ضرورة وضع التوجيه

include < stdio.h >

قبل الدالة ( ) main نظراً لاستخدام الدالتين :

getch ( ) و putchar ( )

الجملة ; (" a\ ") printf تقوم بإصدار صوت الجرس bell تنبيها للخطأ.

6.5 المصفوفات ذات البعدين 2- dimensional arrays

تــتكون المصفوفة ذات البعدين من عدد من الصفوف وعدد من الأعمدة. فمثلا البيانات التالية والتي تخص عدد الطلبة في 4 فصول دراسية و 3 أقسام بمعهــد عالٍ ، يمكن تمثيلها بمصفوفة ذات بعدين :

الفصل الدراسي	1	2	3	4
طلبة الحاسوب	78	56	42	35
طلبة الرياضة	58	47	61	38
طلبة الاحصاء	45	36	25	19

الشكل (6.5.1) أعداد الطلبة حسب الفصل الدراسي والتخصص

إذا رمزنا لهذه المصفوفة بالاسم students ، يمكننا الإعلان عنها في لغة سى على الصورة :

```
int  students [3][4]  ;
```

حيث عدد الأعمدة =4 و عدد الصفوف=3 .

ومعنى ذلك أن students [0][0] تعنى عدد طلبة الحاسوب (computer) في الفصل الأول ، أما students[1][1] فتعنى عدد طلبة الرياضة (Math) في الفصل الثاني ، وهكذا .....

<u>مثال</u> : اكتب برنامجا لحساب وطباعة :

أ. مجموع عدد الطلبة في كل قسم.
ب. مجموع عدد الطلبة في كل فصل .

علماً بأن المعطيات كما هو مبين بالشكل (6.5.1) .

البرنامج المطلوب مبين بالشكل (6.5.2) ، دعنا نلقى نظرة فاحصة على هذا البرنامج، لتستوقفنا النقاط التالية :

```
#define N 3
#define M 4
main()
{
        int  students[3][4]
              ={78,58,45,56,47,36,42,61,25,35,38,19}
```

```
             , dept[3], sem[4],    i, j;
        char format[10]=" %d\t";
        for ( i=0 ; i<N ; i++)
        {       dept[i]=0;
                for(j=0; j<M ; j++)
                        dept[i] += students[i][j];
        }
        for(j=0; j < M ; j++)
        {       sem[j]=0;
                for(i=0 ; i<N ; i++)          sem[j] += students[i][j];
        }
        for(i=0 ; i<N; i++)
        {       printf("\n") ;
                for(j=0 ; j<M; j++)
                        printf(format, students[i][j])  ;
                printf(" %d",dept[i]);
        }
        printf("\n");
        for(j=0; j<M ; j++)          printf(format,sem[j]);
}
```

الشكل(6.5.2) برنامج معالجة مصفوفة ذات بعدين

① نلاحظ أنه لا توجد دالة الإدخال scanf في البرنامج ، ولكن تم وضع الجدول المطلوب داخل البرنامج عند الإعلان عن المصفوفة students على النحو التالي:

```
int studens [3][4] = { 78 , 58 , 45 , 56 , 47 , 36 ,
                       42 , 61 , 25 , 35 , 38 , 19 }
```

حيث وضعنا الجدول على شكل قائمة أفقية ابتداء من العمود الأول في الجدول ، يليه العمود الثاني ، .... وهكذا .
تسمى هذه الطريقة بتخصيص القيم الابتدائية للمصفوفة array initialization .
ويمكن استخدامها في المصفوفة الأحادية وثنائية البعد على حد سواء . على سبيل المثال الجملة :

```
char  format [10]  = " %d \ t "  ;
```

تقوم بالإعلان عن أن المتغير format هو من نوع النضيد ، وتخصص له القيمة الابتدائية " %d \ t " .

② الوصف " %d \ t " يعنى طباعة عدد صحيح ثم ترك بعض الفراغات نظراً لوجود الرمز t \ الذي يستخدم في طباعة الجداول tab . أي أن تأثير t \ هو الانتقال إلى حقل جديد في نفس السطر.

وبالمناسبة فإن الرموز التالية تستخدم عادة في عملية الوصف :

الرمز	المعنى
\a	الجرس bell

\b	الرجوع إلى الـــوراء خانـــة واحـــدة back space
\n	الانتقال إلى سطر جديد
\t	الانتقال إلى حقل جديد في نفس السطر

6.6 ترتيب الأسماء

نظراً لما لعملية ترتيب قوائم الأسماء ترتيبا أبجديا من أهمية بالغة في التطبيقات الإدارية، نخصص لها هنا موضوعا كاملا.

نحتاج طبعا عند ترتيب الأسماء إلى مقارنتها أبجديا. فإذا قارنا مثلا بين الاسم " Ahmed" والاسم " Ali "
نجد أن

"Ali " > "Ahmed "

رغم أنهما يتساويان في الحرف الأول ، إلا أن الحرف الثاني من "Ali" يـــأتي في الترتيب بعد الحرف الثاني من " Ahmed" .

تفيدنا في هذه المقارنـــة الـــدالة strcmp التـــي تقـــوم بمقارنـــة نـــضيدين ، ويأتي هذا الاسم من : string comparison ، وهى تعمل على النحو التالي:

جامعة الفاتح –كلية العلوم – طرابلس – ليبيا
omarzarty@yahoo.com

$$\text{strcmp}(\text{s1}, \text{s2}) = \begin{cases} negative & if \quad s1 < s2 \\ 0 & if \quad s1 = s2 \\ positive & if \quad s1 > s2 \end{cases}$$

وبما أننا نريد أن يكون الترتيب تصاعديا (أي من الأسماء التي تبدأ بحرف A إلى الأسماء التي تبدأ بحرف Z) سيكون الاختبار على النحو التالي:

```
if ( strcmp (name[i+1] , name[i] ) < 0 )
     { sorted = 0 ;
       strcpy ( temp , name [i] ) ;
       strcpy (name [i] , name [i+1] ) ;
       strcpy ( name [i+1] , temp ) ;
     }
```

نلاحظ هنا استغلال الدالة strcpy في نسخ نضيد إلى آخر، حيث لا يجوز في لغة سى أن نكتب مثلاً :

temp = name [i] ;

لأن temp و name [i] من نوع النضيد string .

اتبعنا في هذا البرنامج نفس الخوارزمية التي اتبعناها في البرنامج بالشكل (6.3.1) مع ملاحظة الآتي :

❶ استخدام المصفوفة ثنائية البعد name[N][L]، حيث N عدد الأسماء بالقائمة و L طول كل اسم بالحرف .

❷ استخدام الدالة strcpy لنسخ نضيد إلى آخر .

❸ استخدام الدالة strcmp لمقارنة نضيدين .

```
#define  N 5
#define  L 12
main()
{       char name[N][L], temp[L] ;
        int i ,j, k  , sorted;
        for(i=0; i<N ; i++)
        {
                printf("\nenter name[%d] ",i);
                scanf("%s",&name[i]);
        }
        for(k=0; k<N; k++)
        {       sorted=1;
                for(i=0; i<N-1 ; i++)
                {
                 if(strcmp(name[i+1] , name[i])<0 )
                        {       sorted = 0;
                                strcpy(temp, name[i] );
                                strcpy(name[i] , name[i+1]);
                                strcpy(name[i+1] , temp);
                        }
                }
                if(sorted) break;
        }
        printf("\n Here is the sorted array after %d iterations",k);
        for(i=0; i<N ; i++)
```

```
        printf("\n %s",name[i]);
}
```

الشكل (6.6.1) برنامج ترتيب الأسماء

مثال : إضافة اسم إلى قائمة مرتبة

نفرض أن لدينا المصفوفة old_array مرتبة ترتيباً أبجديا تصاعديا، و أن المطلوب هو كتابة برنامج لإضافة اسم ووضعه في المكان المناسب من القائمة الجديدة new_array .

تسمى هذه العملية بالإدراج inserting ، وهى تتطلب (بعد تحديد المكان المناسب للاسم الجديد) إزاحة كل الأسماء التي تليه في القائمة. فمثلا إذا كانت القئمة القديمة مرتبة على النحو التالي:

i	Old Array
0	Anas
1	Huda
2	Lubna
3	Omar
4	Suad

.وأضفنا إليها الاسم Shada فإن القائمة الجديدة يجب أن تكون على النحو التالي:

i	New Array
0	Anas
1	Huda
2	Lubna
3	Omar
4	Shada
5	Suad

و لتحديد الموقع (وليكن m) للاسم المضاف new_name ، نلاحظ أن :

old_array [m] < new_name < old_array [m+1]

وهذا هو الشرط الذي نستخدمه في البرنامج لتحديد قيمة m .

والآن يـــتضح أن المصفوفة الجديدة new_array يمكن تكوينها كما يلى :

```
if  i < m     new_array[i] = old_array[i];
if  i == m    new_array[i] = new_name;
if  i  > m    new_array[i] = new_array[i-1];
```

حيث N .... , 3 , 2 , 1 , 0 =i

والشكل (6.6.3) يبين البرنامج المطلوب ، مع ملاحظة الآتي :

المصفوفة الجديدة new_array تزيد عن المصفوفة القديمة old_array مـــن حيث الحجم بموقع واحد ، وبالتالي فإن الإعلان عن المصفوفتين يتم على النحو التالي:

char old_array [N][L] , new_array [N+1][L] , new_name[L] ;

لتحديد موقع new_name نجرى الاختبار:

old_array[i] < new_name < old_array[i+1]

```
#define  L 12
#define  N 5
main()
{       char old_array[N][L], new_array[N+1][L] ,
        new_name[L];
        int i ,m  ;
        for(i=0; i<N ; i++)
        {
                printf("\nenter name[%d] ",i);
                scanf("%s",&old_array[i]);
        }
        printf("\nPlease enter new name → ") ;
        scanf("%s",new_name);
        for(i=0;i<N; i++)
                if( strcmp( new_name , old_array[i]) > 0 &&
                   strcmp( new_name , old_array[i+1]) < 0 )
                {  m=i+1 ;  break ;  }
```

```
        for(i=0; i<=N ; i++)
    { if(i<m)
           strcpy(new_array[i],old_array[i]);
         if(i==m)
           strcpy(new_array[i], new_name);
         if(i>m)
        strcpy(new_array[i], old_array[i-1]);
        }
        printf("\n Here is the new array ");
        for(i=0; i<=N ; i++) printf("\n %s",new_array[i]);
}
```

الشكل(6.6.3) إضافة اسم إلى قائمة مرتبة.

ويكتب في لغة سى على النحو التالي :

```
     strcmp ( new_name   , old_array [i] )  >  0
&&  strcmp (new_name  ,  old_array [i+1] )  <  0
```

نلاحظ أن إضافة اسم واحد للمصفوفة أدت إلى تكوين مصفوفة أخرى جديـــدة ، وهذا يعتبر إسرافاً في استغلال الذاكرة . لذلك نستخدم ما يعرف عادة بـــالقوائم المرتبطة linked lists التي تعتمد على مفهوم المؤشرات pointers وهو ما سندرسه فيما بعد .

كما نلاحظ أن المصفوفة old_array والمصفوفة new_array ذواتا بعد ثابت ، يتم تحديده في جملة الإعلان عن النوع والبعد ، في بداية البرنامج . ولكن من الناحية العملية من الأفضل أن تكون هناك إمكانية تغيير حجم المصفوفة (أي بُعد المصفوفة) حسب مقتضيات الأمر. فقد نحجز في الذاكرة مواقع أكثر مما نحتاج، مما يسبب ضياع مواقع التخزين ، أو قد نحجز أقل مما نحتاج مما قد يسبب فشل البرنامج . هذه المشكلة أيضاً يتم حلها بواسطة المؤشرات ، التي نتناولها في باب خاص.

6.7 تمارين

1. اكتب برنامجا لحساب مجموع مربعات الأخطاء

$$s = \sum (x_I - xbar)^2$$

حيث $\sum$ تعني الجمع من I=0 إلى I=n-1. أما xbar فهي ترمز لمتوسط قيم x_I.

افترض أي قيمة صحيحة للمتغير n.

لماذا نحتاج إلى استخدام المصفوفة في هذا البرنامج ؟

2 . المعادلة

$$y_{i+1} = (1+r) y_i$$

تصف الرصيد في المصرف بعد مرور i من السنوات، حيث r يمثل النسبة المئوية للربح. اكتب برنامجا لحساب :

أ. الرصيد بعد 10 سنوات.
ب. مقدار الربح بعد 10 سنوات.
ج. عدد السنوات اللازمة حتى يفوق الربح الرصيد الابتدائي.

اعتبر أن الرصيد الابتدائي هو 1000 وأن $r = 0.05$.

3. اكتب برنامجا لقراءة مصفوفة من 10 عناصر عددية صحيحة، و اختبار ما إذا كانت المصفوفة مرتبة تصاعدياً .

4. اكتب برنامجا يقوم بقراءة عدد كسري ويتحقق من عدم وجود خطأ في الإدخال وذلك بعدم قبول أي رمز غير الأرقام من 0 إلى 9 والنقطة العشرية.

5 . اكتب برنامجا يقوم بقراءة نضيد مكتوب بالحروف الصغيرة

a , b , c , ... z

وتحويلها إلى حروف كبيرة

A , B , C , ... Z

جامعة الفاتح –كلية العلوم – طرابلس – ليبيا
omarzarty@yahoo.com

لاحظ أن الفرق بين الحرف الصغير ومقابله من الأحرف الكبيرة هو مقدار ثابت (في جدول أسكى) وهو 32. فمثلاً رقم c في جدول أسكى هو 99، بينما رقم C هو 67.

6. لديك مجموعة من الطلبة ودرجاتهم في مقرر البرمجة. والمطلوب ترتيب أسمائهم حسب درجاتهم ، أي أول اسم في القائمة هو صاحب أعلى درجة ، وهكذا ..

7 . إذا كانت البيانات x_i مرتبة تنازلياً أو تصاعديا ، وكان عددها فرديا ، فإن القيمة الوسطى تسمى (الوسيط) ، أما إذا كان عددها زوجيا فإن الوسيط هو نصف مجموع القيمتين الوسطيين .
اكتب برنامجا لحساب الوسيط لكلتا الحالتين .

8. اكتب برنامجا يقوم بقراءة مصفوفتين ذواتا N صف M عمود ويطبع مجموعهما.

9. اكتب برنامجا يقرأ مصفوفة a ذات N صف و M عمود، والمصفوفة b ذات M صف و L عمود، ويقوم بحساب المصفوفة c حيث

$$c = a*b$$

وذلك حسب التعريف التالي:

$$c_{ij} = \sum a_{ik} \quad b_{kj}$$

حيث $\sum$ ترمز لعملية الجمع من k=1 إلى k=M ، وعلما بـأن الـمـصفوفة الناتجة من حاصل الضرب تتكون من N صف و L عمود .

10 . اكتب برنامجا يقوم بقراءة نضيد string وحرف c وعدد صـحيح m ، حيث m أقل من طول النضيد ، ويقوم البرنامج بإدراج insert الحرف c فـي الموقع m من النضيد . مثال : النضيد " university " والحرف e يدرج فـي الموقع 4 من النضيد فيصبح "university " .

11. أعد البرنامج في تمرين (10) ولكن لغرض إلغاء حرف بدلاً من إضـافة حرف.